Corel® Paradox® 8

December Newsletter December 1, 1997

Internet Publishing with Corel® Paradox® 8

Corel® Paradox® 8 Internet tools include several components that help end-users and developers publish Corel Paradox 8 tables, reports and forms to the Internet and import HTML content to Corel Paradox 8 tables. Internet publishing uses the service of the Corel Web Server, a component available in Corel Paradox 8. Data can be published to the Web using Experts as well as through a programmatic interface using the HTML publishing library.

Basic Publishing

There are two ways to publish Web documents: Static and Dynamic.

Static Web publishing converts the content you want to publish to HTML. Data is retrieved from a database and placed in an HTML file. Static documents represent snapshots of the data content at the time of publishing.

Dynamic Web publishing automatically builds a template of the report or table when it is published. All necessary information about the report or table is stored in the Corel Web Server Repository. Dynamic documents are produced on request and contain the most up-to-date information at the time of request.

> *Corel Web Server* acts as an intermediary between Web browsers and compatible applications (such as IntraBuilder and Corel® Paradox®) to transfer requests and responses between them. It checks whether the requested published document is static or dynamic, handles the static documents and gets a request handler to process the dynamic requests. It's available as an executable as well as an ActiveX control.

Publishing tables and reports

Corel® Paradox® 8 simplifies the process of publishing to the Web by providing easy to use Experts. Tables and reports can be published statically and dynamically.

Publishing Tables

1. Open the table.
2. Click File, Publish to HTML.
3. Follow the steps in the HTML Table Expert†.

The HTML Report Expert helps you convert a report to an HTML text file so that you can publish the report on the Web. The Expert adds the appropriate HTML tags and parameters automatically. You can modify the resulting HTML document as you would any other HTML document.

> † In the final step, the expert asks you how the document should be published and prompts you for a filename to be given to the HTML document. Corel Web Server is case-sensitive, so make sure you note the exact filename used when saving documents. If you choose static publishing, HTML files are created in the directory set by the Base path configuration parameter. If you choose dynamic publishing, a template for the document is created and stored along with other related information in the Web repository. A physical file is not created. It is generated on the fly at the time of request.

Using Corel Web Server

You must have a TCP/IP network connection for the Corel Web Server to function properly.

Setup

Choose Corel Web Server from the Start menu. By default, it is placed in the Accessories submenu of the Corel® Paradox® 8 program group. The Corel Web Server icon appears on the Windows® 95 Taskbar. It is active and ready to receive browser requests.

Configuration

Right-click the Corel Web Server icon to open a menu with options for viewing connections and settings properties.

Server properties
: control how the Corel Web Server handles connections.

Pages properties
: indicate the location of Web pages accessed by the Corel Web Server.

Logging properties
: indicate whether you want to keep log files of connections and transactions and, if so, the names of the log files.

MIME properties
: are records in the Corel Web Server MIME Types database.

Status properties
: control client identification detail and status display.

For details about the properties, refer to Appendix A.

Advanced Publishing

About Web Servers and HTTP

There are two players involved: Web browsers & Web servers.

Typically, the browser or Web client requests a document from the server and it is displayed. The transfer of documents takes place using HTTP. If the user activates a link in the HTML document, the Web client retrieves and displays the linked document.

HTTP is a stateless request/response protocol. A client sends a request to the server consisting of the request method and document identifier. This is followed by a MIME-like message containing request modifiers and other attributes. The server responds with a status line, including the message's protocol version and a success or error code. This is followed by a MIME-like message containing server information and possible entity-body content.

- Sample Request/Response

Request

Client ---------- > Server

GET /customer.html HTTP/1.0
Accept: text/html, image/gif, image/jpeg,, */*
User-Agent: Mozilla/3.01Gold (WinNT; I)
...

Response

Client < ---------- Server

HTTP/1.0 200 OK
Date: 06/17/1997 2:28:09 PM
LastModified: 06/17/1997 2:28:12 PM
Server: CWS/1.0b3
MIME-version: 1.0
Content-type: text/html
Content-length: 2345
<HTML><HEAD><TITLE> . . . </TITLE>...

CGI scripts in a nutshell

An HTML document can contain forms which are presented as a collection of fillout fields and an 'action' associated with the form. This 'action' is a reference to a script on the server side that is executed when triggered, typically using a button. When the user finishes filling in the details, he clicks the button triggering the action. The Web client sends this information to the Web server. The server starts the script, the response is prepared, and it is sent back to the client.

- Example of Form

```
<HTML>
<HEAD>
<TITLE>Simple Input Form</TITLE>
</HEAD>

<BODY>
<H1 ALIGN=CENTER>
<B>Enter Your Name</B></H1>
<P><HR></P>

<P><FORM ACTION="FormData1" METHOD="POST">
<B>Name:</B><BR>
<INPUT NAME="Name"></P>

<P><INPUT TYPE="SUBMIT"
VALUE="Submit"></FORM></P>

</BODY>
</HTML>
```

This HTML form looks like this:

The Method token indicates the method to be performed on the resource identified by the Request-URI and can typically be one of GET/POST/HEAD.

Depending on which method you use, the CGI script will receive data from the HTML document in a different way.

The GET method

If your form has METHOD="GET" in its FORM tag, your CGI program will receive the encoded form input in the environment variable QUERY_STRING.

The POST method

If your form has METHOD="POST" in its FORM tag, your CGI program will receive the encoded form input on stdin.

The HEAD method

The HEAD method is identical to the GET method, except that the server/CGI program DOES NOT return a message-body in the response. (i.e., only 'header' information is requested). This method is often used for testing hypertext links for validity, accessibility, and recent modification.

Publishing a Form

Corel® Paradox® 8 can also publish a form to a static HTML document. This feature is useful for creating an exact replica of your Corel Paradox 8 form for use on the Internet, eliminating the need to recreate it in an HTML editor. This feature works best with simple forms that use text, edit boxes, list boxes, radio buttons or check boxes. Form objects such as graphics, table frames, crosstabs, notebooks and charts do not translate statically to HTML.

- Publishing a form to an HTML file
- Processing information from HTML input forms

Publishing a form to an HTML file

1. Open the form.
2. Click File, Publish to HTML.
3. Type a filename.
4. Choose .HTM as the file type from the drop-down list.
5. Click the Save button.

Corel Paradox 8 saves an HTML version of your form that can be viewed by a Web browser.

Note
Corel Paradox 8 automatically adds FORM METHOD and ACTION tags to any form published to HTML. By default the FORM METHOD tag is set to POST and the ACTION is set to the Corel Paradox 8 form object's noise name (e.g., #Form1). To set these properties yourself, change the HTMLMethod property or the HTMLAction property of the form using the Object Explorer.

Corel Paradox 8 also adds a Submit button to static forms published to HTML. For the Submit button to work with the Corel Web Server Control, you must add code to trap the POST action in the OnPostRequest event. For more information, see Processing information from HTML input forms.

Corel Web Server is case-sensitive. Make sure you note the exact filename used when saving documents.

Processing information from HTML input forms

The Corel Web Server Control is an ActiveX control which can be placed in any product that supports OLE controls as a container (e.g., a Corel Paradox 8 form). The container is notified when a Web client requests an HTTP GET, POST or HEAD method. The Corel Web Server Control supplies its container with an OGI (OLE Gateway Interface) event. An OGI event is an ActiveX control ConnectionPoint event. This allows the container to execute an event-handling procedure in its native code so that, like other Corel Paradox 8 objects, it can be programmed using its own set of ObjectPAL properties, methods and events to send custom responses to client requests. The default HTTP response can also be accepted.

You can configure the Corel Web Server Control to trap information submitted by HTML input forms in its OnPostRequest event.

Suppose a Corel Paradox 8 form is published and saved to a file SIMPLEFORM.HTM:

```
<HTML>
<HEAD>
<TITLE>Simple Input Form</TITLE>
</HEAD>

<H1 ALIGN=CENTER><B>Enter Your Name</B></H1>
<P><HR></P>

<P><FORM ACTION="FormData1" METHOD="POST">
<B>Name:</B><BR>
<INPUT NAME="Name"></P>

<P><INPUT TYPE="SUBMIT"
VALUE="Submit"></FORM></P>

</BODY>
</HTML>
```

You can cut and paste the above text into an HTML file to try the following example.

To trap information posted by an HTML input form:

1. Design the Corel Paradox 8 form containing the Corel Web Server Control object.
2. Set the Base Path of the Corel Web Server Control to the form SIMPLEFORM.HTM on the Pages page in the Corel Web Server Control Properties dialog box.
3. Right-click the Corel Web Server Control object and click Object Explorer.
4. Click the Event tab.
5. Double-click the OnPostRequest event to open the Editor window.
6. Add the following code:

```
method OnPostRequest(Request OleAuto, Response
OleAuto)
var
      vname        string
endvar
      if request.URI="/FormData1" then
          vname=request.getfield("Name")

          Response.ResultString = string(
          "<HTML><H1>Thank You ", vname,
"!</H1></HTML>" )
      endif
endMethod
```

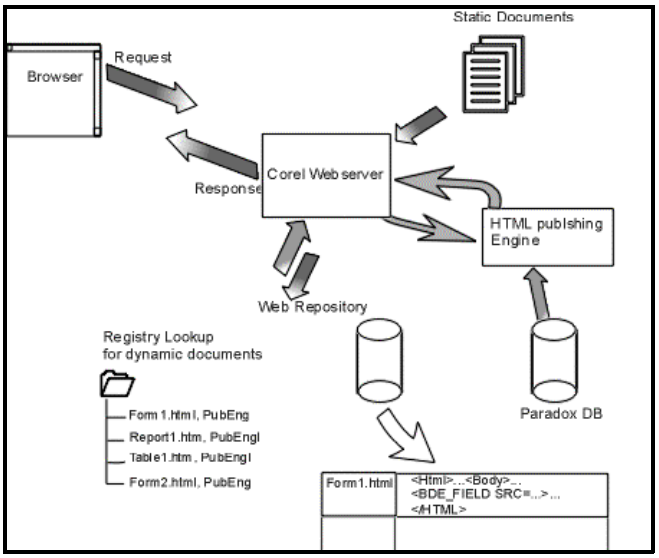
You can request SIMPLEFORM.HTM from the Corel Web Server Control using your browser. When you click the Submit button, the Corel Web Server Control is prepared to process the information you have typed in the form.

Dynamic Publishing

When a document is published dynamically, an .HTT (HyperText Template) file is generated and stored along with other necessary data in the Web Repository.

An .HTT file is an HTML file with custom metatags. These tags are Corel Paradox-specific, and are used by the HTML Publishing Engine to produce standard HTML files. HTT files can be viewed as HTML in a browser as-is, although the browser will ignore all occurrences of custom tags. Conceptually, an .HTT file is an intermediary step in the Internet publishing process. All layout information is defined but metatags are used in place of tables, queries and other database objects.

HTT files are generally used in dynamic publishing where the layout information is known but the actual data needs to be resolved at run time. The dynamic publishing of Corel Web Server manages .HTT files automatically, so their existence and function is largely transparent to the user. If you are implementing your own dynamic publishing system with the Corel Web Server Control, you can generate .HTT files using File, Publish to HTML for tables, reports and forms.



Architecture Diagram

Function of the Corel Web Server + publishing engine
When a request arrives at the Corel Web Server, it refers to a lookup table to check if it is a dynamically generated document. If not, it looks for the document in the directory given by the Base path, reads it, and sends it back. If it is a dynamic document it gets reference to the publishing engine, its .HTT file, information about its data sources and the type of document. It makes a request block, sets up the publishing engine and dispatches it to the engine. The publishing engine parses the .HTT file, replaces custom tags with actual data content from the data sources and forms a new HTML stream. When it is done, this HTML stream goes back to the Web server. The Web server now sends the response back to the client.

The GxPublish DLL is an OLE Automation server that serves as an HTTP request handler registered in the Corel Web Server Repository. This component produces HTML output dynamically from Corel® Paradox® 8 documents.

GxEngine Template Specification

An .HTT file contains the following custom HTML template tags as defined in the GxEngine Template Specification:

BDE_TABLE	outputs an HTML table containing all of the data in the current SRC data source, including a header and column for each field specified in FLDS
BDE_SCAN	marks a section to be repeated for each record in the current SRC data source
BDE_FIELD	outputs data from SRC field or stored parameter
BDE_GROUP	marks a section to be repeated for each block of records that meets the group criteria
BDE_INDEX	marks a section as an Index (Table of Contents) construct
BDE_BREAK	marks a segment break for an Index (Table of Contents) construct
BDE_INCLUDE	appends another template at the current position

The **BDE_TABLE**, **BDE_SCAN**, **BDE_FIELD**, **BDE_GROUP** tags take a general form:

<BDE_xxx SRC=sourceref FLDS= fieldset *additional attributes...> contained data </BDE_xxx>*

The SRC attribute specifies the data source for this table. It is essentially an identifier for the actual database entity. The Web Repository maintains a map of this source reference and information about it (e.g., database name, table name, type of document, etc.).

The contained data can consist of more BDE_* tags and/or HTML tags to form hierarchical relationships.

For example, a template for a view of a table could be:

```
<BDE_TABLE SRC="CUSTOMER"
FLDS=”Name,Phone” BORDER=2>
```

The output for this would translate to something like this:

```
<TABLE BORDER=2>
<TR><TD>Adam Anderson</TD><TD>867-
5309</TD></TR>
<TR><TD>Billy Watson</TD><TD>431-
1000</TD></TR>

. . . repeat for each record in table . . .

<TR><TD>William Gladstone</TD><TD>1-800-628-
5560</TD></TR>
<TR><TD>Zorba Quincy</TD><TD>879-
2000</TD></TR>
</TABLE>
```

BDE_SCAN repeats the contained data for each record in its data source.

BDE_GROUP takes up a data source and makes available grouping based on various criteria such as number of records to be shown in a row(using FLDS="#" INCR=*n)*, grouping on a field value (using FLDS=fieldset), grouping on n characters of a field(using FLDS=fieldset CHARS=*n),* date values, etcetera.

Consider the following segment:

```
<BDE_GROUP SRC="biolife" FLDS="Category">
   <BDE_SCAN SRC="biolife" FLDS="Common Name"
SORT="+">
        <BDE_FIELD SRC="biolife" FLDS="Species
No">
         <BDE_FIELD SRC="biolife" FLDS="Common
Name">
     </BDE_SCAN>
</BDE_GROUP>
```

It takes data from 'biolife' and places it in groups based on the value in 'category' field. BDE_SCAN takes up the same data source and sorting records within each group and shows two fields: 'Species No' and 'Common Name'.

Using the Publishing Engine

GxPublish DLL
The GxPublish DLL is an OLE Automation server that serves as an HTTP request handler registered in the Corel Web Server Repository. This component produces HTML output dynamically from Corel® Paradox® 8 documents.

GxEngine DLL
This component handles the task of outputting Borland Database Engine (BDE) data in HTML format. It can be

called directly from ObjectPAL or any other language that supports calling OLE Automation servers. It parses HTML documents for BDE-specific template tags and replaces them with snapshots of the data sources.

The GxEngine can be used to convert templates into static HTML documents. The following Methods and Properties can be used to perform template-to-template conversions.

Methods

Execute	generates HTML output using the current templates, data sources, properties and parameters
ExecuteFrom	generates HTML output from a specific position using the current templates, data sources, properties and parameters
AddSourceTable	adds a data source from a table in a directory or BDE alias
RemoveSource	removes a data source from the GxEngine
RefreshSources	updates record buffers in all open data sources
AddLink	adds a link between two data sources
RemoveLink	removes a link between two data sources
AddTemplateFile	adds a template from a file to the GxEngine
AddTemplateString	adds a template from a string to the GxEngine
RemoveTemplate	removes a template from the GxEngine
AddParam	adds a parameter to the GxEngine
RemoveParam	removes a parameter from the GxEngine

Properties

MasterTemplate	the name of the template to process first
IndexTemplate	the name of the default template to use for the BDE_INDEX template tag's template parameter
HeaderTemplate	the name of the default template to use for the BDE_INDEX template tag's head parameter
FooterTemplate	the name of the default template to use for the BDE_INDEX template tag's foot parameter
OutputMethod	specifies whether the conversion should output to a file (1) or string (2)
OutputFileName	specifies the name and destination for the output file. Used only if OutputMethod = 1
OutputString	contains the HTML output generated by calling Execute or ExecuteFrom. Used only if OutputMethod = 2
OutputPath	the default location for storing output; the default OutputPath = the path to OutputFileName
ImagePath	the default location for storing binary image output; the default ImagePath = OutputPath
PrivateDir	directory to use for all temporary or auxiliary files; the default = Windows temporary directory
LongFileNames	specifies whether the directory in OutputFileName supports long filenames
Dynamic	indicates whether the document to be published is dynamic or static
NumberFormat	specifies the default format for all integer conversions
FloatFormat	specifies the default format for all floating-point conversions
CurrencyFormat	specifies the default format for all currency conversions
DateFormat	specifies the default format for all date conversions
TimeFormat	specifies the default format for all time conversions
DateTimeFormat	specifies the default format for all date-time conversions

There is a sequence of steps to be followed.

Typically, the following sequence of steps should be followed:

1. An instance of the GxEngine is created by invoking your language's method for creating an OLE Automation object and passing it the server name "Corel.GxEngine"
2. The templates, data sources, and parameters involved in a conversion are first added to an instance of the GxEngine OLE Automation server using the properties and helper methods. Method AddTemplateString specifies the .HTT content that would be parsed. For each source specified in the template, GxEngine has to be supplied with information about the data source. This is done by the method AddSourceTable. The output from this conversion can be stored in a file or returned as reference to a string. You can control this using the OutputMethod property.
3. Then output is generated by "executing" that GxEngine instance (calling the Execute or ExecuteFrom method).

The following script starts the GxEngine, converts the VENDORS.DB table in the working directory to HTML, and saves it as the file VENDORS.HTM in the working directory:

```
method run(var eventInfo Event)
var
    GXEngine OLEAuto
endVar
    GXEngine.Open ("Corel.GXEngine")
```

```
    GXEngine^AddTemplateString
("vendors","<BDE_TABLE SRC=\"VENDORS\"
BORDER=2>")
    GXEngine^AddSourceTable
("vendors",fullName(":WORK:"),"vendors.db")
    GXEngine^OutputMethod = 1          ;output to a file
    GXEngine^OutputFileName = "vendors.htm";output to
the working directory
    GXEngine^Execute()
endMethod
```

Corel® Paradox® 8 HTML Publishing Library

Corel Paradox 8 also provides an easy to use interface to Web publishing functions through an ObjectPAL library, HTMLIB01. To access the library, get the Experts directory using the EspertDir() procedure and then load the HTMLIB01 library.

Overview of the functions

Web publishing functions
There are two high level functions (HTMLPublish_Table and HTMLPublish_Report) to publish tables and reports to the Web. Templates for tables and reports can be generated using two helper functions: GenTemplate_Table, GenTemplate_Report. Both static and dynamic documents can be generated.

Utility functions
The utility methods are used to extract information from the documents being published and to perform various HTML tasks.

Use these methods to get the information you need for use with GXEngine_Execute:
ExtractSourceInfo
ExtractStaticImages

Use this method to find and launch a Web browser:
LaunchBrowser

Use this method to convert color values into HTML text:
RGBtoHTML

Repository functions
The repository methods make up the Repository API, the interface to the Corel Web Server Repository where templatized data is stored for dynamic HTML publishing. These methods let developers fully manipulate the contents of the Repository.

Appendix

Appendix A: Web Server Properties

Appendix B: GxEngine Template Tokens

Server properties

HTTP Port
: The port at which the server listens for connections. No two servers can have the same port on the same machine. Values are 1–32767 (Default = 80).

Min Ready Connections
: The number of threads to prepare for connection at startup and keep waiting after startup. The number must be equal to or less than Max Connections. A higher number speeds connections but uses much more memory. Values are 0–255 (Default = 25).

Max Connections
: Total number of connections allowed at one moment. Values are 1–255 (Default = 100).

Connection Timeout
: The number of seconds to wait before issuing a time-out and breaking the connection. This property is used in both reading requests and sending responses. It also specifies how long to keep a keep-alive connection open. Values are 1–600 (Default = 30).

Keep Alive Connections
: Allow Keep Alive Connections. If enabled, a request is permitted to keep the connection open. This allows a browser quicker access on its second request; it doesn't need to wait for a connection to open. Enabling Allow Keep Alive Connections uses up more of the server's available connections.

Max Requests Per Connection

The number of request/response transactions allowed for each keep-alive connection (if Allow Keep Alive Connections is enabled). Values are 1–1024 (Default = 5).

Pages properties

Base Path
The root directory where the Web pages are stored. For security reasons, don't use your own root directory. Choose a directory with subdirectories that users are allowed to access.

Default Page
The default Web page name in each directory.

Footer Page
The full path for an optional page with the information to appear at the bottom of each transferred page.

Header Page
The full path for an optional page with the information to appear at the top of each transferred page.

Logging properties

Access Logging
These settings enable and name a log file of who connected to the server. The access log file is written in Common Log Format.

Enable Access Logging
When enabled, logs connection history to a file in Common Log Format.

Access File Name
The name of the access log file.

Debug Logging
These settings enable and name a debug log file of debug information for transactions handled by the server.

Enable Debug Logging
When enabled, logs debug information for transactions to a file for later tracing and debugging.

Debug File Name
The name of the Debug Log file where transaction records are stored for later tracing and debugging. The log includes the buffer received from the browser, the buffer sent to the browser and a dump of the variables from both the request and response.

MIME properties

Add
Displays the New MIME Type dialog box where you can enter a file extension with associated media type and subtype.

Extension
The extension used for a certain media type (for example, TXT).

MIME Type
The main media type associated with an extension (for example, TXT).

MIME Subtype
The category of the main media type associated with an extension (for example, plain).

Edit
Displays the Change MIME Type dialog box where you can change the MIME Type or MIME Subtype associated with the selected extension.

Remove
Deletes the selected extension's record.

Status properties

Enable Client Name Lookup
When enabled, attempts to convert an IP address to an Internet machine name. This task takes more time than when Enable Client Name Lookup is disabled, but more connection information is obtained.

Enable Server Status Bar
When enabled, the server status bar shows information about the current status of the server and its connections. Enable Server Status Bar must be checked before these updates are made; but the updates are independent of other properties.

Show Active Connections
When enabled, active threads appear in the server list (state = New/Recv, Processing, Sending).

Show Completed Connections
When enabled, completed thread connections (status = Dead) appear in the server list.

Number of Completed Connections

If Show Completed Connections is enabled the number of connections with "status = Dead" are displayed in the Corel Web Server connection list.

BDE_TABLE

Syntax

<BDE_TABLE SRC=sourceref [FLDS=fieldset] [ENCODE=onoff] [INDEX=fieldset] [tabletags ...]>

Parameters

SRC	Data source to scan
FLDS	Field(s) to output (default: all fields)
ENCODE	Encode the field output changing all illegal characters to URL escape sequences (for example, change "#" into "%42")
INDEX	Fields to be used in the generated INDEX (if nested in BDE_INDEX template tag)
tabletags	Any additional tags to be placed in the HTML table declaration <TABLEtabletags>

BDE_SCAN

Syntax

<BDE_SCAN SRC=sourceref FLDS=fieldset SORT="+,-,...">html</BDE_SCAN>

Parameters

SRC	Data source to scan
FLDS	Fieldset to use for grouping criteria (same as BDE_GROUP)
SORT	Sort order for each field in FLDS; + = ascending and - = descending

BDE_FIELD

Syntax

<BDE_FIELD {[SRC=sourceref] [FLDS=fieldset] [INDEX=fieldref] [FORMAT=formatspec] | PARAM=fieldref} [ENCODE=onoff]>

Parameters

SRC	Data source (default: current innermost SRC)
FLDS	Field(s) to output (SRC=tableref - default: Blank) (SRC=groupref - default: "LOW")
INDEX	Values to be used in the generated INDEX (if in BDE_INDEX template tag)
FORMAT	Standard Delphi format string (applies to Date, Time, DateTime and Numeric type fields)
PARAM	Output stored parameter value
ENCODE	Encode the field output changing all illegal characters to URL escape sequences (for example, change "#" into "%42")

BDE_GROUP

Syntax

<BDE_GROUP [NAME=groupref] [SRC=sourceref] FLDS=fieldset [{CHARS=integer [INCR=integer] | DATE=dateval | INCR=integer }] [SORT=sortref]> html </BDE_GROUP>

Parameters

NAME	Name to give the temporary data source this group creates
SRC	Data source to group (default: current innermost SRC)
FLDS	Field to use for grouping criteria (only the first field in the list is used for grouping; any additional fields are used to sort results)
CHARS	Number of characters to group by, starting with the first character
DATE	Type of date grouping (DAY, WEEK, MONTH, QUARTER, YEAR)
INCR	Range to consider in each group; not used when grouping on date fields INCR used with CHARS indicates ordinal character value to break (for example, FLDS=”Name” CHARS=1 INCR=5 outputs group:A-E, group:F-J, etc...)
SORT	Sorting method to use for the first field; additional fields listed in FLDS will be sorted ascending (default: NONE)

BDE_INDEX

Syntax

<BDE_INDEX
[NAME=indexref][TEMPLATE=templateref]
[FILES=breakref] [HEAD=templateref]
[FOOT=templateref]>html</BDE_INDEX>

Parameters

NAME	Name to give the temporary data source this index creates
TEMPLATE	Template name to construct INDEX (default: HTML Engine IndexTemplate property)
FILES	SINGLE: Index contents appended to single file MULTI: Index contents appended to single file; new file generated per sub-index ATBREAK: Index contents separated into separate files at each <BDE_BREAK> marker (default: SINGLE)
HEAD	Header template to use for generated files (default: GxEngine's HeaderTemplate property)
FOOT	Footer template to use for generated files (default: GxEngine's FooterTemplate property)

BDE_BREAK

Syntax

<BDE_BREAK>

BDE_INCLUDE

Syntax

<BDE_INCLUDE TEMPLATE=templateref>

Parameters

TEMPLATE	Template name to insert